Sicurezza e autenticazione nei dispositivi

Prof. Massimiliano Sala

Università degli Studi di Trento, Lab di Matematica Industriale e Crittografia

Trento, 7 Maggio 2012

1. Il problema crittografico

2. Crittografia pubblica/ Crittografia simmetrica

3. Chiavi deboli

Il problema crittografico

Introduzione

Obiettivo crittografico classico:

Consentire a due utenti di comunicare su un canale **potenzialmente insicuro**, senza permettere ad **una terza persona** di comprendere il contenuto dei messaggi.

Impedire ad una **terza persona** di impersonare uno dei due utenti.


Alice


Bob


Eve

Crittografia a chiave privata

Possiamo pensare l'insieme dei messaggi da trasmettere come l'insieme delle stringhe binarie lunghe n.
Formalmente $\mathcal{M} = \mathbb{F}^n$, dove $\mathbb{F} = \{0, 1\}$.

Una **funzione di cifratura** è una mappa $f : \mathcal{M} \to \mathcal{M}$ (bigettiva) che trasforma plaintext in ciphertext, in modo che a ciascun plaintext corrisponda un solo ciphertext e viceversa.

In linea di principio Alice e Bob si mettono d'accordo sulla funzione di cifratura da usare.

Una funzione $f : \mathcal{M} \to \mathcal{M}$ in realtà è un algoritmo che **deve** essere complesso per non essere ricostruibile ("attaccabile").
Sarebbe troppo oneroso mettersi d'accordo sull'intero algoritmo, quindi Alice e Bob si mettono d'accordo su un algoritmo che dipende da un parametro chiamato **chiave**.

Crittografia a chiave privata

Esiste una chiave K, che si usa sia per cifrare che per decifrare.

Scelta una chiave K una **funzione di cifratura** è una mappa $f_k : \mathcal{M} \to \mathcal{M}$ (bigettiva) che trasforma plaintext in ciphertext, in modo che a ciascun plaintext corrisponda un solo ciphertext e viceversa.

Alice e Bob possono pubblicare f e scambiarsi segretamente solo K.

Robustezza del sistema crittografico

Solitamente lo spazio delle chiavi $\mathcal{K} = \mathbb{F}^k$, cioè una chiave è un vettore di k bit. L'azione del cifrario deve apparire completamente casuale.

$$\begin{array}{ccc} 1 & \to & m \\ 2 & \to & ? \\ 3 & \to & ?? \\ \dots & \dots & \dots \end{array}$$

Quindi l'unico modo per rompere un cifrario ideale è provare tutte le chiavi, che costa

$$2^k.$$

Useremo k per confronto (es. per il caso ideale $k{=}k$, in generale $k \geq k$). Ad esempio se un sistema ha un insieme di $2^{10} = 1024$ chiavi, ma si rompe usando solo 2^6 cifrature, allora si dice che:
$k = 10$
$k = 6$.

Principali cifrari

I principali cifrari usati sono:

3DES	$k = 168$	$k = 113$
AES256	$k = 256$	$k = 254$
RC4	$k = 256$	$k = 256$ (?)
AES128	$k = 128$	$k = 126$

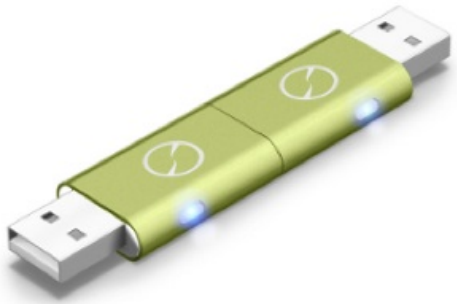
RC4 bisogna usarlo con attenzione.

Come scambiarsi la chiave in maniera sicura?

- non la scambiamo!
- la scambiamo su un canale fisicamente protetto

CONFIDACE

SATA



iTwin

Chi parla con il dispositivo?

La chiave sta su un dispositivo.

Dove sta anche?

- La chiave sta anche in un altro dispositivo
- La chiave sta in un server remoto.

Chi sono io?

Se la chiave sta su un dispositivo, chi lo prende decifra a volontà.

Per evitare questo ci sono due soluzioni:

- mettere come password, *passphrase* che sblocca il token,
- mettere un check biometrico sul token (es: impronte digitali).

Se non ci incontriamo?

1. Come posso dimostrare a qualcuno che io sono chi dico di essere?
2. Come faccio a scambiare fisicamente la chiave se la persona è lontana?

Due problemi che hanno la stessa soluzione:
la crittografia pubblica

Crittografia a chiave pubblica

Non esiste solo una chiave privata, perchè siamo costretti ad usare un canale pubblico.

Diffie-Hellman

che consente a due entità di stabilire una chiave condivisa e segreta utilizzando un canale di comunicazione pubblico.

Cifratura asimmetrica

grazie alla quale si cifra e decifra usando una combinazione di chiave pubblica e privata (non condivisa).

Diffie-Hellman

L'algoritmo di DH richiede che Alice  e Bob  si creino dei valori segreti random e permette di arrivare ad un segreto condiviso (chiave).

L'attacco migliore **Function Field Sieve**, ha complessità

$$2^{n^{\frac{1}{3}}(\log_2(n))^{\frac{2}{3}}}$$

$k \sim \sqrt[3]{n}$, ma per $80 \leq k \leq 256$ si ha $n \sim k^{1.7}$.

RSA

L'algoritmo di cifratura asimmetrica RSA si basa sulla difficoltà del seguente problema.
Siano p e q numeri primi, sia $N = pq$, dato N trovare p e q.

$$[n = \log_2(N)]$$

Il miglior algoritmo è una versione del General Number Field Sieve, che gira con complessità

$$2^{n^{\frac{1}{3}}(\log_2(n))^{\frac{2}{3}}}$$

$k \sim \sqrt[3]{n}$, ma per $80 \leq k \leq 256$ si ha $n \sim k^{1.7}$.

Quindi RSA fornisce in pratica la stessa (in)sicurezza di DH.

Chi sono io?

Un grosso vantaggio del meccanismo di chiave pubblica e privata è che è possibile tenere una lista (PKI/CA) di chiavi pubbliche liberamente consultabile da tutti.

Facendo una *challenge* con la tua chiave privata dimostri chi sei.

Challenge

Questo **non** lo puoi fare con la simmetrica perchè dovresti svelare la tua chiave. Con la chiave pubblica invece:

- X crea una stringa random s e te la manda
- tu cifri s con la tua chiave privata, ottieni c e la mandi a X
- X decifra c con la tua chiave pubblica e ritrova s

Siamo alle solite, chi ha la tua chiave privata è te.

La cosa più comoda è tenerla su un dispositivo.

Firma e altro

Nel momento in cui la tua identità è certa, quella viene usata per firmare ufficialmente certi documenti e per identificare che certe operazioni le hai fatte tu.

Questi aspetti verranno approfonditi nell'ultimo intervento.

Crittografia pubblica/ Crittografia simmetrica

Scontro tra titani

Spesso si discute se sia meglio l'uso della crittografia pubblica o simmetrica.
Considerando che attualmente si usano gli algoritmi con i seguenti parametri

- AES256 per la crittografia simmetrica
- DH o RSA con 1024 bit (o 2048) per la crittografia pubblica

vediamo cosa possiamo dire sulla sicurezza.

Scontro tra titani

Il miglior attacco noto a AES256 (che recupera la chiave) ha un costo di 2^{254} crittazioni.
Supponendo che

- un core medio riesca a eseguire 2^{50} crittazioni al secondo,
- esistano 1000 core ogni abitante del pianeta, compresi neonati e le persone che vivono nella giungla amazzonica,
- al mondo ci siano 20 miliardi di persone,

Allora **TUTTA la potenza del mondo unita** dovrebbe lavorare senza sosta per 10^{31} MILIARDI di anni.

Scontro tra titani

Usando l'algoritmo GNFS si riescono a fattorizzare numeri N dell'ordine di $2^{768} \sim 10^{231}$.

Spendendo un milione di euro per dotarsi di un cluster potente, si potrebbe rompere una chiave RSA-768 (cioé N è dell'ordine di 2^{768}) ogni 6 mesi (in media).

Per cui chiavi RSA-1024 o superiori (purché non siano **chiavi deboli**) sono al momento fuori dalla portata di un attacco pratico.

Scontro tra titani

Sebbene il costo di un attacco RSA1024 sia molto alto (circa un miliardo di euro) non è nulla rispetto al costo astronomico di rompere AES256.

Anche aumentando la lunghezza delle chiavi RSA di qualche migliaio di bit, RSA non si avvicina alla sicurezza fornita da AES256.

Quindi sembrerebbe che la simmetrica sia molto meglio.

Colpi bassi tra titani

Purtroppo c'è molta gente che usa RSA1024/DH per scambiare le chiavi di AES256 e crede di avere la sicurezza di AES256.

Ovviamente la sicurezza è solo quella garantita dall'anello più debole della catena.

Il primo round: la simmetrica

Dal punto di vista puramente crittoanalistico, la simmetrica è nettamente più forte della pubblica, ovvero riesce a garantire una sicurezza molto più elevata con un numero di operazioni paragonabile.

Però la crittografia pubblica sta facendo un lavoro molto più difficile della simmetrica, perchè sta lavorando su un canale pubblico.
Dal punto di vista di teoria dell'informazione, ciò è inevitabile.

Ci sono solo due casi:

- o ci mettiamo in una situazione in cui è possibile scambiarsi fisicamente dei dispositivi (e allora possiamo usare la simmetrica),
- oppure, se non ci scambiamo niente di fisico ma solo elettronico (problemi di distanza, costi, ecc.), dobbiamo usare la pubblica.

Il secondo round: la pubblica

Al giorno d'oggi, la comodità e il bassissimo costo delle comunicazioni digitali ha decretato il sopravvento della pubblica, tranne per quelle organizzazioni che possono permettersi di organizzare (internamente) uno scambio fisico di dispositivi.

Il terzo incomodo: il random

La generazione dei numeri random è fondamentale per la sicurezza, sia in crittografia simmetrica che in quella pubblica:

1. creazione dei parametri privati nella chiave pubblica,
2. creazione dei parametri pubblici nella chiave pubblica,
3. creazione della chiave simmetrica (es. AES256).

Non tutto il random è uguale:

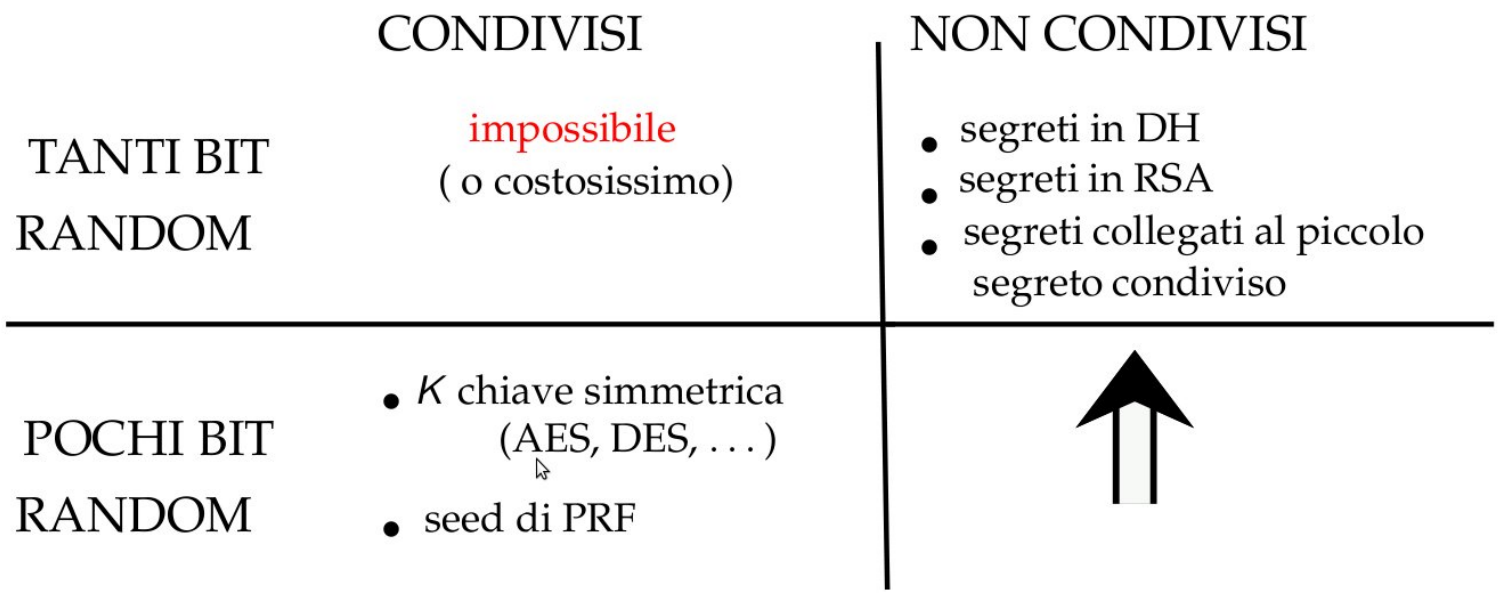
- non c'è necessità di mostrare a nessuno i bit random dei parametri privati;
- questi bit random sono visibili da tutti e questo può essere pericoloso;
- i bit random per creare la chiave simmetrica vanno condivisi in maniera riservata! (Questo è il compito più difficile.)

Creazione randomicità

Esistono principalmente tre modi per ottenere una chiave random:

- utilizzare sorgenti random naturali (es. campionamento del campo elettrico e della voce, rilevamento fenomeni quantistici)
- utilizzare algoritmi pseudo-random (PRF)
- utilizzare una commistione di sorgenti random e pseudo-random

Quanti random?



PRF: seed$\rightarrow$ 101000111 . . . . . .

Aspetti della randomicità

Esistono essenzialmente due aspetti fondamentali che riguardano la randomicità in quest'ambito:

1. creazione di sequenze pseudo-random da parte dei dispositivi
2. generazione (pseudo-random) dei seed.

Per tali ragioni si ricorre all'uso di PRF (Pseudo Random Function).

Purtroppo **non è possibile dimostrare** matematicamente la (pseudo-)randomicità di un algoritmo.

Sequenze pseudo-random

Per una buona PRF deve valere:

1. Per ogni SEED s e COUNTER c (usato), anche se l'attaccante conosce $f(s,1),\ldots,f(s,c)$ **non** è in grado di trovare $f(s,c+1)$.
   - 1-a in particolare, Π deve essere grande per ogni s
   - 1-b anche conoscendo $f(s,1),\ldots,f(s,c)$ **non** si può trovare s
2. idealmente anche se l'attaccante conosce il seed s, non deve essere in grado di recuperare il counter c da $f(s,c)$.

Generazione seed (pseudo-random)

Non solo è importante che le sequenze siano generate pseudo-random, ma **a maggior ragione** che la distribuzione dei seed sia il più possibile random
Ciò che **non vogliamo** è che

$$\text{conoscere } s_1,\ s_2, \ldots, s_{i-1} \implies \text{conoscere } s_i.$$

Per questo la distribuzione dei seed dovrebbe essere il più possibile vicino all' **entropia pura**. Ciò si può ottenere mediante generatori hardware di randomicità che sfruttano fenomeni quantistici quali:

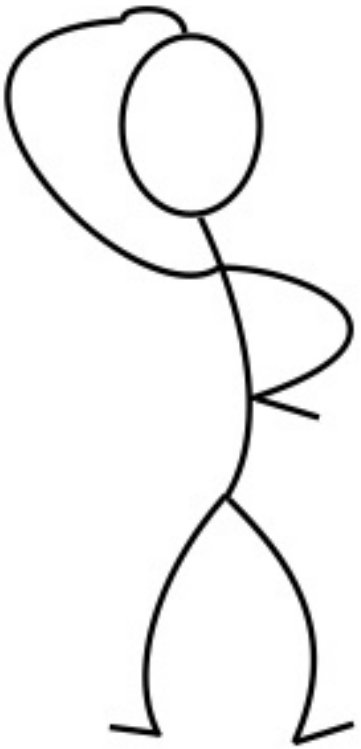
- variazione del campo elettro-magnetico
- effetto fotoelettrico

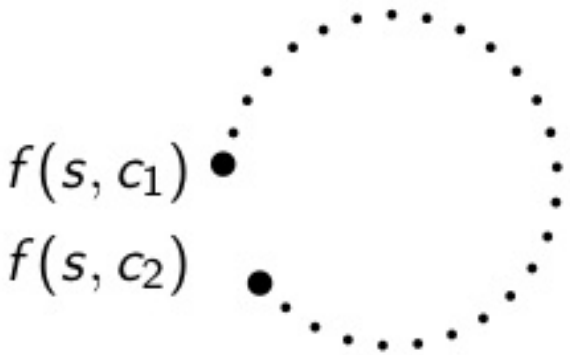
PCI Express

PCI

Il punto di vista dell'osservatore esterno

$$
\begin{array}{rcl}
f(s, c_1) & \rightarrow & 101011011110 \\
f(s, c_2) & \rightarrow & 101010001110 \\
\vdots & & \vdots \\
f(s, c_i) & \rightarrow & 101110011001 \\
f(s, c_{i+1}) & \rightarrow & ?? \\
& \searrow & \\
& & s?? \\
\end{array}
$$





Chiavi deboli

Chiavi deboli

Una chiave debole, K, in un sistema crittografico (sia pubblico che privato) è una chiave che permette la rottura del sistema.

Invece, esistono **numerose chiavi deboli** sia per DH ed RSA.

Riteniamo quindi fondamentale per la sicurezza valutare attentamente la robustezza delle chiavi usate (e quindi la loro generazione)

Quanto deboli le deboli?

Le seguenti chiavi RSA 1024:

- $N = 313256375876377212440015318323108678520006173008034636063302469362957128066154999976915271523113164049627718408844530703484989822647023510143641161477819303748834981751956165887681798612125473753385171353058142794346935884074455798705326089160080254660362883833231925389841878822795695169771993196329575554015$
  $e = 29$
- $N = 703606728829425222386754525735867532353561794249517072279524442134219740346812313911122376577798291308343173684600484528948470625632095674346511670057314481944944614776388555094301742500113459518088266875640891456768243692575911682249129488446432250644872863589855553619827615234588027031314797877915$
  $e = 37$

Le ho date come sfida ai team di studenti impegnati nelle CryptoWars e le hanno rotte in

Quanto deboli le deboli?

Le seguenti chiavi RSA 1024:

- $N = 313256375876377212440015318323108678520006173008034636063302469362957128066154999976915271523113164049627718408844530703484989822647023510143641161477819303748834981751956165887681798612125473753385171353058142794346935884074455798705326089160080254660362883833231925389841878822795695169771993196329575554015$
  $e = 29$
- $N = 703606728829425222386754525735867532353561794249517072279524442134219740346812313911122376577798291308343173684600484528948470625632095674346511670057314481944944614776388555094301742500113459518088266875640891456768243692575911682249129488446432250644872863589855553619827615234588027031314797877915$
  $e = 37$

Le ho date come sfida ai team di studenti impegnati nelle CryptoWars e le hanno rotte in **meno di un secondo**.

Hanno rotto anche una chiave di curve ellittiche da 300 bit in **meno di un minuto**!

Ma tutto questo è solo un gioco?

Presentiamo i risultati di un recentissimo lavoro scientifico appena uscito (14/02/2012), che porta la firma di Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter.

- 270.000 siti che condividono la chiave pubblica con **almeno** un altro sito;
- 71.052 chiavi sono usate da tanti siti diversi (stessa chiave anche per **migliaia** di siti);
- 12.720 chiavi RSA1024 sono vulnerabili e si rompono in **meno di un millisecondo**.

Ancora sul random

Ancora sul random

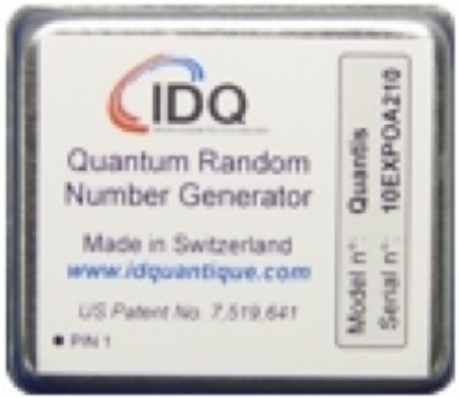
Un modo per ottenere random è usare dei parametri biometrici:

- impronta digitale
- faccia o altre immagini (riprese di telecamera da smartphone)
- firma biometrica

Ancora sul random

Si potrebbe creare del random ideale usando la meccanica quantistica ma nessuna sorgente fisica di randomicità è stata approvata dal NIST, perchè nella fabbricazione con la tecnologia attuale non si riescono ad evitare imperfezioni.
Quindi i generatori fisici attuali basati sulla meccanica quantistica fanno un post processing dei bit generati fisicamente.

USB



OEM

PCI Express

PCI

Grazie per l'attenzione